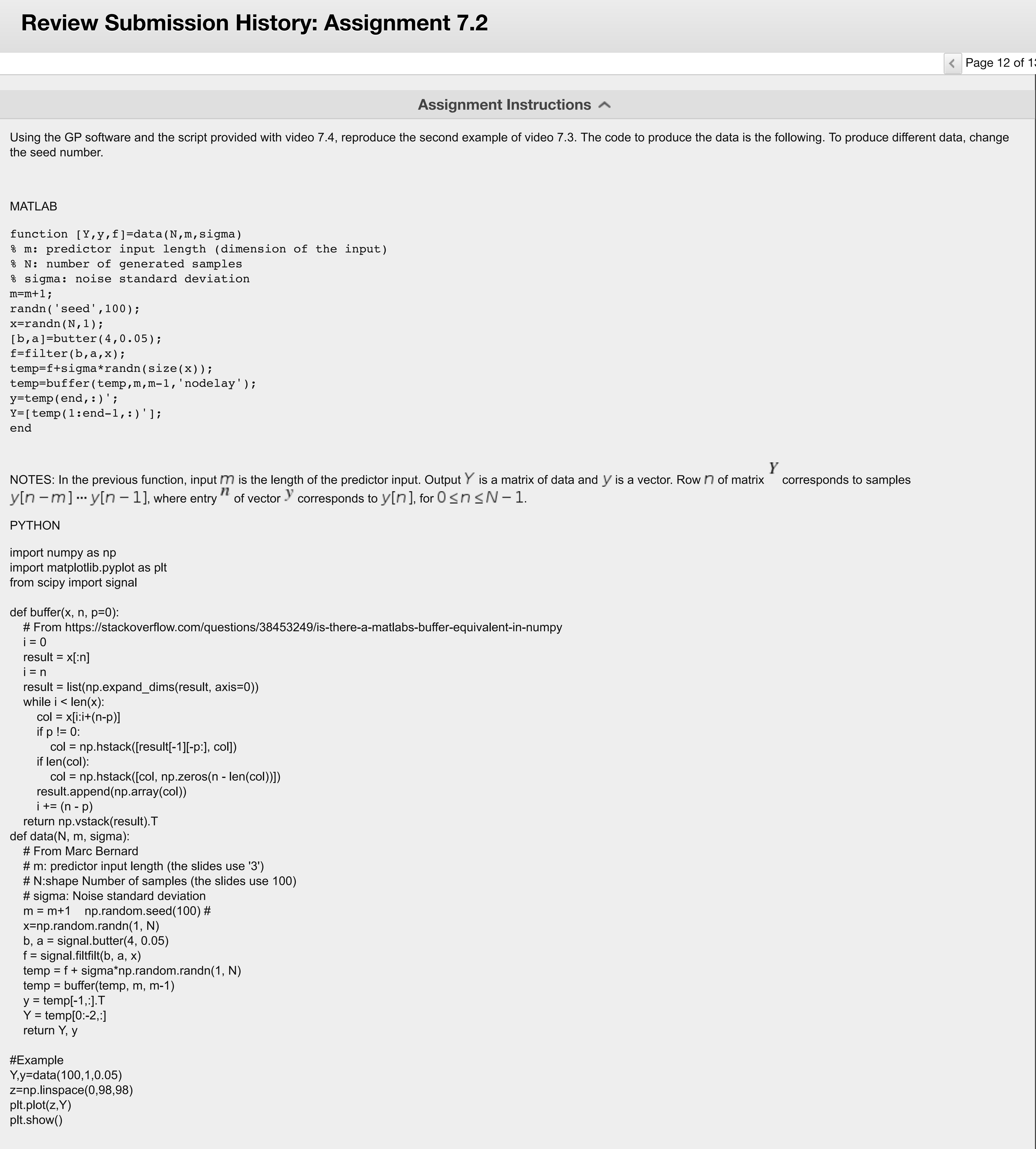

Review Submission History: Assignment 7.2

Assignment Instructions

Using the GP software and the script provided with video 7.4, reproduce the second example of video 7.3. The code to produce the data is the following. To produce different data, change the seed number.

MATLAB

```
function [Y,y,f]=data(N,m,sigma)
% m: predictor input length (dimension of the input)
% N: number of generated samples
% sigma: noise standard deviation
m=m+1;
randn('seed',100);
x=randn(N,1);
[b,a]=butter(4,0.05);
f=filter(b,a,x);
temp=f+sigma*randn(size(x));
temp=buffer(temp,m,m-1,'nodelay');
y=temp(end,:)';
Y=[temp(1:end-1,:)'];
end
```

NOTES: In the previous function, input m is the length of the predictor input. Output Y is a matrix of data and y is a vector. Row n of matrix Y corresponds to samples $y[n-m] \cdots y[n-1]$, where entry n of vector y corresponds to $y[n]$, for $0 \le n \le N-1$.

PYTHON

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

def buffer(x, n, p=0):
    # From https://stackoverflow.com/questions/38453249/is-there-a-matlabs-buffer-equivalent-in-numpy
    i = 0
    result = x[:n]
    i = n
    result = list(np.expand_dims(result, axis=0))
    while i < len(x):
        col = x[i:i+(n-p)]
        if p != 0:
            col = np.hstack([result[-1][-p:], col])
        if len(col):
            col = np.hstack([col, np.zeros(n - len(col))])
        result.append(np.array(col))
        i += (n - p)
    return np.vstack(result).T
def data(N, m, sigma):
    # From Marc Bernard
    # m: predictor input length (the slides use '3')
    # N:shape Number of samples (the slides use 100)
    # sigma: Noise standard deviation
    m = m+1    np.random.seed(100) #
    x=np.random.randn(1, N)
    b, a = signal.butter(4, 0.05)
    f = signal.filtfilt(b, a, x)
    temp = f + sigma*np.random.randn(1, N)
    temp = buffer(temp, m, m-1)
    y = temp[-1,:].T
    Y = temp[0:-2,:]
    return Y, y

#Example
Y,y=data(100,1,0.05)
z=np.linspace(0,98,98)
plt.plot(z,Y)
plt.show()
```